PATENT
Docket No. **SYB/0114.00**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Sudipto R. Chowdhuri	Examiner: Morrison, Jay A
Serial No.: 10/711,931	Art Unit: 2168
Filed: October 13, 2004	APPEAL BRIEF
For: Database System with Methodology for Parallel Schedule Generation in a Query Optimizer	

Mail Stop Appeal
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

BRIEF ON BEHALF OF SUDIPTO R. CHOWDHURI

This is an appeal from the Final Rejection mailed July 23, 2008, in which currently-pending claims 1-15, 17, 19-40, 43-62 and 64-70 stand finally rejected. Appellant filed a Notice of Appeal on November 26, 2008. This brief is submitted electronically in support of Appellant's appeal.

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

The real party in interest is assignee Sybase, Inc. located at One Sybase Drive, Dublin, CA 94568.

2. RELATED APPEALS AND INTERFERENCES

There are no appeals or interferences known to Appellant, the Appellant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

The status of all claims in the proceeding is as follows:

Rejected: 1-15, 17, 19-40, 43-62 and 64-70

Allowed or Confirmed: None

Withdrawn: None

Objected to: None

Canceled: 16, 18, 41, 42, 63

Identification of claims that are being appealed: Claims 1-15, 17, 19-40, 43-62 and 64-70

An appendix setting forth the claims involved in the appeal is included as Section 8 of this brief.

4. STATUS OF AMENDMENTS

Four Amendments have been filed in this case. Appellant filed an Amendment on January 3, 2007 in response to a non-final Office Action dated October 3, 2006. Subsequently, Appellant filed an Amendment in conjunction with a Request for Continuing Examination on October 22, 2007 in response to a final Office Action mailed on April 6, 2007. Additionally, in response to a non-final Office Action issued on January 7, 2008, Appellant filed an Amendment on May 7, 2008. In the Amendment filed on May 7, 2008, the pending claims were amended in a manner that Appellant believes clearly distinguished the claimed invention over the art of record, for overcoming the art rejections. In response to the Examiner's Final Rejection dated July

23, 2008 (hereinafter "Final Rejection") finally rejecting Appellant's claims, Appellant filed a Notice of Appeal. Appellant also filed an Amendment After Final on January 13, 2009 to address an Examiner objection to one of Appellant's claims as containing informalities. However, Appellant has chosen to forego making further amendments to the claims as Appellant believes further amendments are not warranted in view of the art.

5. SUMMARY OF CLAIMED SUBJECT MATTER

Appellant asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 1:** In a database system (see e.g., Appellant's specification paragraph [13], paragraphs [79]-[91]; also see generally e.g., Fig. 3) a method for parallel optimization of a query (see e.g., Appellant's specification Abstract, paragraphs [13]-[15], paragraph [95], paragraph [132]; see generally Fig. 6, Figs. 7A-B), the method comprising: generating a plurality of parallel plans for obtaining data requested by the query (see e.g., Appellant's specification paragraph [13], paragraph [132] (generates a set of plans); Fig. 6 at 610 (output set of plans)), the parallel plans including parallel operators for executing portions of the query in parallel (see e.g., Appellant's specification paragraph [13], paragraphs [103]-[106]; paragraph [132]; Fig. 5 at 502-503 and 505-507), adjusting parallel operators of each parallel plan based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable (see e.g., Appellant's specification paragraph [13], paragraphs [134]-[135], paragraphs [139]-[146], paragraphs [155]-[157]; Fig. 7A at 702-704; Figs. 9A-B at 901-909; Fig. 12A at 1206, 1207), creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan (see e.g., Appellant's specification paragraph [13], paragraph [15], paragraph [113], paragraphs [131]-[132], paragraph [134], paragraphs [136]-[138], paragraphs [151]-[157]; Fig. 6 at 620; Figs. 7A-B at 705-714; Figs. 12A-B), wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query (see e.g., Appellant's specification paragraphs [13]-[15], paragraphs [131]-[132], paragraph [134], paragraphs [136]-[138], paragraphs [151]-[157]; also see generally e.g., Fig. 6 at 620; Figs. 7A-B at 705-714; Figs. 12A-B), and

includes separating a resource intensive operator into a plurality of operators (see e.g., Appellant's specification, paragraph [136], paragraphs [155]-[156]; see also, e.g., Figs. 7A-B at 705-712, Fig. 12A at 1207), determining execution cost of each parallel plan based on its schedule (see e.g., Appellant's specification, Abstract, paragraph [13], paragraph [15], paragraph [114], paragraphs [132]-[134], paragraph [137]; Fig. 7B at 713-714), and returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query (see e.g., Appellant's specification paragraph [13], paragraph [15], paragraph [138]; Fig. 6 at 620 (best plan with schedule output); Fig. 7B at 715).

Appellant additionally asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 26:** A system for parallel optimization of a database query (see generally e.g., Appellant's specification, paragraph [14], paragraphs [94]-[95], paragraph [132]; also see generally, e.g., Fig. 6, Figs. 7A-B), the system comprising a search engine for generating a plurality of parallel plans which can be used for obtaining data requested by the query (see e.g., Appellant's specification, paragraph [14], paragraph [61], paragraph [129], paragraph [132] (generates a set of plans); Fig. 6 at 610 (search engine outputs set of plans)), the parallel plans including parallel operators for executing portions of the query in parallel (see e.g., Appellant's specification, paragraph [14], paragraphs [103]-[106]; paragraph [132]; also see e.g., Fig. 5 at 502-503 and 505-507); a parallel scheduler for adjusting parallel operators of each parallel plan based on maximum number of threads available for executing the query (see e.g., Appellant's specification, paragraph [14], paragraphs [134]-[135], paragraphs [139]-[146], paragraphs [155]-[157]; Fig. 7A at 702-704; Figs. 9A-B at 901-909; Fig. 12A at 1206, 1207) and creating a schedule for the parallel plan indicating a sequence for execution of operators of the parallel plan (see e.g., Appellant's specification, paragraph [14], paragraph [113], paragraphs [131]-[132], paragraph [134], paragraphs [136]-[138], paragraphs [151]-[157]; Fig. 6 at 620; Figs. 7A-B at 705-714; Figs. 12A-B), wherein the maximum number of threads is user configurable (see e.g., Appellant's specification paragraph [13], paragraphs [134]-[135], paragraphs [139]-[146], paragraphs [155]-[157]; Fig. 7A at 702-704; Figs. 9A-B at 901-909; Fig. 12A at 1206, 1207) and wherein the

schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query (see e.g., Appellant's specification, paragraphs [13]-[15], paragraphs [131]-[132], paragraph [134], paragraphs [136]-[138], paragraphs [151]-[157]; also see generally e.g., Fig. 6 at 620; Figs. 7A-B at 705-714; Figs. 12A-B), and the parallel scheduler separates a resource intensive operator into a plurality of operators (see e.g., Appellant's specification, paragraph [136], paragraph [156]; see also, e.g., Figs. 7A-B at 705-712, Fig. 12A at 1207), and a module for determining execution cost of each parallel plan based on its schedule and available resources (see e.g., Appellant's specification, paragraph [14], paragraph [114], paragraphs [132]-[134], paragraph [137]; Fig. 7B at 713-714), and returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query (see e.g., Appellant's specification, paragraph [14], paragraph [138]; Fig. 6 at 620 (best plan with schedule output); Fig. 7B at 715).

Appellant further asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 48:** A method for parallel optimization of a query requesting data from a database (see e.g., Appellant's specification Abstract, paragraphs [13]-[15], paragraph [132]; see generally Fig. 6, Figs. 7A-B), the method comprising creating a plurality of operator trees for executing the query (see e.g., Appellant's specification paragraph [15], paragraph [132] (generates a set of plans (operator trees)); Fig. 6 at 610 (output set of plans)), the operator trees providing for execution of portions of the query in parallel (see e.g., Appellant's specification paragraph [15], paragraphs [103]-[106]; paragraph [132]; Fig. 5 at 502-503 and 505-507); adjusting the portions of the query to be executed in parallel based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable (Appellant's specification, paragraph [13], paragraph [15], paragraph [132], paragraphs [140]-[146]; also see e.g., Fig. 7A at 703, Figs. 9A-9B), generating a schedule for execution of each operator tree based upon dependencies among operators of each operator tree and resources available for executing the query (see e.g., Appellant's specification, paragraph [13], paragraph [15], paragraphs [131]-[132], paragraph [134], paragraphs [136]-[138], paragraphs [151]-[157]; Fig. 6 at 620; Figs. 7A-B at 705-714;

Figs. 12A-B) including separating a resource intensive operator into a plurality of operators (see e.g., Appellant's specification paragraph [136], paragraphs [155]-[156]; see also, e.g., Figs. 7A-B at 705-712, Fig. 12A at 1207), and returning a result indicating the operator tree having lowest execution cost based on its schedule for executing the query with available resources (see e.g., Appellant's specification paragraph [13], paragraph [15], paragraph [138]; Fig. 6 at 620 (best plan with schedule output), Fig. 7B at 713-715).

Appellant additionally argues based on **dependent claims 10, 35 and 56** which include claim limitations pertaining to: generating a parallel plan using a partitioning property so as to partition data among operators of the parallel plan (see e.g., Appellant's specification, paragraph [55], paragraph [116], paragraph [132]; see generally, Fig. 6 at 610; Fig. 7A at 701). .

Appellant additionally argues based on **dependent claims 11, 36 and 57** which include claim limitations of generating a cost vector for each operator tree (see e.g., Appellant's specification, paragraph [113], paragraph [120], paragraph [132]; Fig. 6 at 610; Fig. 7A at 701).

Appellant also argues based on **dependent claims 12, 37 and 58** which includes claim limitations providing that the cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator, and resource usage of an operator for a certain time period (see e.g., Appellant's specification, paragraphs [114]-[115], paragraphs [123]-[129]).

Additionally, Appellant argues based on **dependent claims 13, 38 and 59** which includes claim limitations of pruning a first operator tree having a cost vector costing more in each vector dimension than a second operator tree (see e.g., Appellant's specification, paragraph [113], paragraph [129]).

6. GROUNDS OF REJECTION TO BE REVIEWED

The grounds for appeal are:

(1st) Whether claims **1-15, 17, 19-40, 43-62 and 64-70** are unpatentable under 35 U.S.C. Section 103(a) as being obvious over Srivastava, et al "Optimizing Multi-Join Queries in Parallel Relational Databases", in Proceedings of the Second International Conference of Parallel and Distributed Information Systems, Los Alamitos, California,

December 1993 (hereinafter "Srivastava") in view of U.S. Patent 7,047,530 of Lu (hereinafter "Lu"), and further in view of U.S. Published Application 20050125427 of Dageville et al (hereinafter "Dageville").

7. ARGUMENT

A. First Ground: Claims 1-15, 17, 19-40, 43-62 and 64-70 rejected under 35 U.S.C. 103(a)

1. General

Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie case of obviousness under this section, the Examiner must establish: (1) that there is some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). The reference(s) cited by the Examiner fail to meet these conditions.

2. Claims 1-9, 14-15, 17, 19-34, 39-40, 43-55, 60-62 and 64-70

The Examiner has rejected Appellant's claims 1-15, 17, 19-40, 43-62 and 64-70 under 35 U.S.C. Section 103(a) as being obvious over Srivastava, et al "Optimizing Multi-Join Queries in Parallel Relational Databases", in Proceedings of the Second International Conference of Parallel and Distributed Information Systems, Los Alamitos, California, December 1993 (hereinafter "Srivastava") in view of U.S. Patent 7,047,530 of Lu (hereinafter "Lu"), and further in view of U.S. Published Application 20050125427 of Dageville et al. (hereinafter "Dageville"). The following rejection of Appellant's claim 1 by the Examiner is representative of the Examiner's rejection of the Appellant's claims under Section 103 based on Srivastava, Lu and Dageville:

> As per claim 1, Srivastava teaches
> In a database system, a method for parallel optimization of a query, the method

comprising: (see abstract)
generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel; (query plan space, section 4.1, page 87, first paragraph)
adjusting parallel operators of each parallel plan; (for each tree and all subtrees, section 4.1, page 87, first paragraph)
creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan, wherein the schedule is created based upon dependencies among operators of each parallel plan (ordered tree where shape represents intra-operator parallelism, section 2, page 85, first paragraph) and resources available for executing the query; (query optimization considers resources available, section 6, page 91, first paragraph)
determining execution cost of each parallel plan based on its schedule. (combining operator costs, section 3.2, page 87)
and returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query. (query plan representation for expressing intra and inter-operator parallelism, processor and memory assignment, and execution time estimate, section 6, page 91, first paragraph)

Srivastava does not explicitly indicate "based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable".
However, Lu discloses "based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable" (maximum number of threads, column 6, lines 23-27; maximum threads configured appropriately, column 8, lines 5-9; column 15, lines 25-32).
It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Srivastava and Lu because using the steps of "based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable" would have given those skilled in the art the tools to improve the invention by allow parallel compilation using makefiles. This gives the user the advantage of being able to control aspects of the compilations via the makefile.

Neither Srivastava nor Lu explicitly indicate "and includes separating a resource intensive operator into a plurality of operators".
However, Dageville discloses "and includes separating a resource intensive operator into a plurality of operators" (rewrite high-load sql statement into semantically equivalent form, paragraph [0010], lines 1-5).
It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Srivastava, Lu and Dageville because using the steps of "and includes separating a resource intensive operator into a plurality of operators" would have given those skilled in the art the tools to improve the invention by ensuring that resources are not needlessly wasted on sql statement which can be reworked. This gives the user the advantage of more efficient use of computer resources.

(Final Rejection, paragraph 4, pages 2-5)

At the outset, it is important to understand that Appellant does not purport to have invented the broad notion of executing query plans in parallel. A number of different approaches exist for generating query plans, including parallel query plans, and choosing the best plan for executing a given query. Therefore, the general notion of executing query plans in parallel is understood in the industry and documented in prior art references such as Srivastava. However, the focus of Appellant's invention is a specific improvement for optimizing query plans that may execute in parallel, by creating a schedule for each parallel plan based on available resources and then using this schedule to compare costs of alternative plans and select the best plan. Appellant's approach of generating a plurality of different parallel plans, adjusting the plans to account for available resources, creating a schedule for execution of each of these plans (including introducing additional parallelism when doing so will improve performance), and determining the best parallel plan amongst those plans differ substantially from prior art solutions. The particular features which distinguish Appellant's invention from the prior art of record are set forth as claim limitations in Appellant's claims and are discussed in detail below.

By way of background, selecting the best plan for execution of a query in a parallel processing environment is a complex task as the underlying SQL operations may be performed in a multitude of different ways. For example, a parallel processing system could scan a given table serially or in parallel. As the scan is being performed in parallel, the result set from the scan could be merged into one stream that is funneled into a distinct operation or the data streams could be re-split into a larger number of sub-streams so that the distinct operation can be done with a greater degree of parallelism. Similar choices also need to be made for various other operations, including whether to perform a grouping operation serially or in parallel, and so forth and so on. Appellant's invention addresses these complexities by providing for creation of a schedule for each parallel plan indicating the sequence for execution of operators of each parallel plan and associated costs in a manner that is not taught or suggested by the prior art.

Appellant's solution generates a plurality of parallel plans, structured as trees of

operators, for executing a given query (see e.g., Appellant's specification paragraph [13], paragraph [105]; paragraph [132]; also see e.g., Fig. 5 at 501-507). The operators included in such trees refer to constructs that reflect SQL operations, such as join operations, predicate evaluation mechanisms, scans (e.g., table or index), grouping operations, union operations, distinct operations -- that is, operations defined by SQL. Appellant's claimed invention provides for examining these operations to determine whether performing particular operations in parallel will yield improved performance in executing the query. A particular focus of Appellant's invention is identifying the best possible parallel plan for execution of a query given the resources that are available for executing the query.

Srivastava discusses representing a parallel query plan as a "capacitated labeled ordered binary tree" (Srivastava, Section 2, page 85). However, as discussed below in greater detail, Srivastava indicates that only queries consisting of join operations are considered. Additionally, Srivastava states that developing a parallel cost model is a major task that is beyond the scope of the paper (Srivastava, Section 3.1, page 86). Appellant's solution specifically attempts to address some of these complex problems that were recognized (but not solved) by Srivastava by examining operators in an operator tree and the available resources to determine whether performing particular operations in parallel will yield improved performance (i.e., reduce execution costs) in executing the query in a given environment.

Appellant's solution includes features for taking into account the resources which are available and adjusting the parallel plans that are generated to account for the resources (e.g., memory and processors) available for executing the query. Appellant's approach considers two general kinds of resources during schedule generation that are referred to as preemptable resources (PS) and non-preemptable resources (NPS). More particularly, Appellant's invention uses the concept of multidimensional PS and NPS resources in developing a cost model with sufficient flexibility for parallel query execution (see e.g., Appellant's specification paragraphs [118]-[120]). Query operators are represented as pairs of vectors with one dimension per PS and NPS resource, respectively. Preemptable resources include disks, CPU, network interfaces, and the like (see e.g., Appellant's specification, paragraph [118]). Non-preemptable resources include

memory buffers, whose time-sharing among operators introduces prohibitively high overheads (see e.g., Appellant's specification, paragraph [119]). Thus, the inclusion of NPS requirements gives rise to trade-offs. In some instances increasing the degree of parallelism of an operator reduces the NPS requirement, while decreasing it allows the parallelism to be kept fairly coarse-grained so that the communication cost does not overwhelm the execution (Appellant's specification, paragraph [120]).

Appellant's solution provides for specifying and maintaining an NPS capacity requirement throughout the query execution process (see e.g., Appellant's specification, paragraph [120]). More particularly, during the scheduling process, a determination is made as to whether the maximum NPS resource usage of a pipeline exceeds the maximum NPS resource that is available (see e.g., Appellant's specification, paragraphs [155]-[156]; Fig. 12A at 1206). If the NPS resource usage of the pipeline is greater than the NPS resource available, operators are added to materialize the pipeline into a plurality of pipelines, each of which are within the NPS capacity constraints (Appellant's specification, paragraph [156], Fig. 12A at 1207). In other words, Appellant's claimed invention provides for by introducing additional parallelism into a plan based on applicable resource constraints. These distinctive features are included as limitations of Appellant's claims. For example, Appellant's amended claim 1 includes the following claim limitations:

> In a database system, a method for parallel optimization of a query, the method comprising:
> generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;
> adjusting parallel operators of each parallel plan based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable;
> creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan, wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query and includes separating a resource intensive operator into a plurality of operators;
> determining execution cost of each parallel plan based on its schedule; and
> returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query.

(Appellant's amended claim 1, emphasis added)

As illustrated above, Appellant's scheduling process includes examining dependencies among operators of the plans and resource constraints and costs. For each operation Appellant's solution may decide to perform the operation serially or in parallel. Therefore, there are a wide range of possibilities available in how serial and parallel operations (including different levels of parallelism within a given operation) are combined. There is also another dimension of choices available in how a given operation is performed. For example, a scan operation may be performed using a table scan or an index scan (as available). For a distinct operation, the system may perform a hash-based distinct operation or sort-based distinct operation. Similarly, for grouping, the system could do a hash-based grouping or sort-based grouping, or if the data is already sorted within the system, it could perform the grouping on the already-sorted data. As should be readily apparent, combining these dimensions of possibilities in all possible ways yields *n* different plans, where *n* may be a very large number.

For all of those possible plans, Appellant's system provides for computing a cost to determine which of the possible plans is the best plan. In determining the cost (elapsed time) for each possible plan, Appellant's system determines how all the various operators are to be scheduled. As previously described, resource constraints and costs are considered in the process of creating and evaluating a schedule. The schedule indicates exactly when an operator would start and when would it stop, based on available resources (Appellant's specification, paragraph [136]; Figs. 7A-B at 705-712). Based on that information, that is the activation schedule of various operators, Appellant's system can determine the elapsed time for each particular plan (operator tree), and thus can determine which plan is in fact the best one. If the elapsed time of the current operator tree being considered is less than the minimum cost (i.e., elapsed time) of any previously evaluated operator tree, the current operator tree is saved as the best plan (best operator tree) and the elapsed time of the current plan is made the new minimum (Appellant's specification, paragraph [137]; Fig. 7B at 713-715).

The Examiner references Srivastava for the corresponding teachings of generating a schedule for execution of each operator tree based on available resources. However, Srivastava acknowledges that because of the "wide variety of parallel architectures

available", developing analytical cost expressions for them is a "major task" beyond the scope of the Srivastava paper (Srivastava, Section 3.1, page 86). Thus, Srivastava makes various assumptions about the queries that are received, including that the queries consist only of join operations and that there is enough main memory to hold the smaller relation of all joins (Srivastava, Section 3.1, page 86). Thus, while Appellant's scheduling methodology includes evaluation of the actual resources that are available, Srivastava simply assumes that sufficient resources are available and admits that it cannot handle queries when this assumption does not hold true (Srivastava, Section 3.1, page 86). In the Final Rejection (paragraph 6 at page 11), the Examiner contends that Srivastava (at paragraph 6 on page 91) includes teachings of considering resources that are available. However, the referenced portion of Srivastava merely states that the search space needed for query optimization on a parallel machine is much larger than that required for sequential cases as follows:

> Query optimization for parallel machines needs to consider machine architecture, processor and memory resources available, and different types of parallelism, making the search space much larger than the sequential case. Minimizing *parallel execution time*, rather than *work*, however, creates the following circular dependence: a plan tree is needed for effective resource assignment, which is needed to estimate the parallel execution time, and this is needed to guide the cost-based search for a good plan tree. An exhaustive search breaks this cycle since it enumerates all possible trees. In this paper we proposed a new search heuristic which breaks the cycle by constructing a plan tree layer by layer in a bottom up manner. Lower and upper bounds on execution time for plans consistent with the decisions made so far are estimated and used to guide the search....
>
> (Srivastava, paragraph 6, page 91, emphasis added)

Respectfully, the above teachings of Srivastava simply represent a problem statement recognizing that optimization of queries in a parallel processing environment requires a larger search space. However, Srivastava indicates that addressing this problem requires a solution for breaking the "circular dependence" of needing a plan tree for resource assignment in order to estimate execution time and needing this information in order to guide a cost-based search for a good plan for executing the query. Appellant's invention provides a solution which addresses this problem by creating a schedule for

each parallel plan being considered indicating a sequence for execution of operators of each parallel plan based on available resources, determining execution time of each plan based on its schedule and selecting plans having favorable execution costs. Srivastava uses a different approach to break the "circular dependence" problem that is described above. Basically Srivastava's solution provides for making certain assumptions (including, as noted above, that there is enough main memory to hold the smaller relation of all joins) and then uses a heuristic so as to examine a much smaller portion of the search space (see e.g., Srivastava, paragraph 4, page 87). Thus, while Srivastava recognizes the problem area addressed by Appellant's invention, Srivastava does not provide a solution comparable to that of Appellant's claimed invention. In particular, Srivastava does not create a schedule for each query plan based on available resources, nor does Srivastava determine the cost of each alternative plan based on the schedule in order to determine which plan is, in fact, the best for execution of a particular query in a given environment.

Additionally, Srivastava provides no teaching of separating a resource intensive plan operator into a plurality of operators as provided in Appellant's specification and claims. As noted above, Srivastava simply assumes memory is sufficient and does not evaluate whether or not sufficient memory is available for performing a given operation. Thus, it is clear that Srivastava's teachings are not comparable to Appellant's claim limitations of evaluating available resources (e.g., memory) and separating a resource intensive operator into a plurality of operators when doing so will provide improved performance. In the Final Rejection, the Examiner acknowledges that Srivastava does not include teachings of separating a resource intensive operation into a plurality of operators and therefore adds Dageville as providing such teachings.

Turning to Dageville, one finds that Dageville describes a "tuning advisor" solution that generates recommendations to a user for tuning particular database queries (Dageville paragraph [0012]). For example, the recommendations generated by Dageville's tuning advisor can include path analysis to check for missing indices, statement structure analysis to check for badly written statements, and statistics analysis to check for missing or stale data statistics. The specific teachings referenced by the Examiner in the Final Rejection (at paragraph 4 on page 5 of the Final Rejection) discuss

a toolkit for rewriting a given SQL statement into sematically equivalent forms (Dageville paragraph [0010]). However, Dageville makes no mention of parallel query plans or of separating a resource intensive operator into a plurality of operators so as to perform such operation in parallel as provided in Appellant's specification and claims. Thus, its teachings of rewriting a given SQL statement, which make no mention of performing operations in parallel or evaluating resource requirements for performing operations in parallel are not at all comparable to the specific features of Appellant's claimed invention.

Another significant difference between Appellant's claimed invention and the prior art references is that Appellant's claimed invention provides for adjusting an operator tree based on available threads (worker processes). This illustrates another manner in which Appellant's invention adjusts the degree of parallelism of a plan based on available resources. Here, each of the operator trees (plans) is adjusted for available worker processes (i.e., threads) (see e.g., Appellant's specification, paragraph [134]). This adjustment ensures that an operator tree (plan) does not exceed the maximum number of configured worker processes.

The Examiner acknowledges that Srivastava does not including this teaching of adjusting plans based on maximum threads available for executing the query and therefore adds Lu for these teachings. Lu, however, describes a different parallel processing problem in a very different context; namely, the context of compiling source code of an application that includes thousands of separate source code files (Lu, col. 2, lines 1-2). The particular problem addressed by Lu's invention is that prior make file utilities operate serially even on multi-processor machines (Lu, col. 2, lines 5-13). In this context, Lu provides a command server that is adaptable to provide for an acceptable range of parallel compilation which limits the maximum number of threads that can be spawned for compiling selected targets (Lu, col. 6, lines 18-26). Significantly, Lu does not mention adjusting a query execution plan to ensure that an operator tree (plan) does not exceed the maximum number of configured worker processes. In fact, the Lu reference makes no mention of a query or an operator tree (plan) for executing a query given that it relates to parallel compilation of any application composed of multiple source code files which is a fundamentally different problem than that addressed by

Appellant's claimed invention.

In the Final Rejection (paragraph 7 at pages 11-12), the Examiner argues that Srivastava discloses consideration of the resources available and Lu is added as disclosing that a maximum number of threads to be spawned can be defined. This argument fails for at least the following reasons. First, Srivastava does not, in fact, consider resources (e.g., threads) available in a given environment as discussed in detail above. Secondly, Lu simply limits the number of threads spawned, which is not comparable to Appellant's claimed invention. Appellant's solution operates to adjust parallel operators of parallel plans (see e.g., Appellant's claim 1 set out above) that are created so that such plans do not provide for performing a greater number of operations in parallel than are actually supported by the number of threads available for performing such operations. In other words, Appellant's solution operates to take account the number of threads that available when it creates parallel plans rather than to attempt to adjust the number of available threads. Further distinctions between Appellant's invention and the prior art references area also found in Appellant's dependent claims as discussed below.

3. Claims 10, 35 and 56

Appellant's claims 10, 35 and 56 include claim limitations of generating a parallel plan using a partitioning property so as to partition data among operators of the parallel plan (see e.g., Appellant's specification paragraph [55], paragraph [116], paragraph [132]). At each stage of plan building, the partitioning property provides a pre-computed notion of what partitioning is useful and on what columns (see e.g., Appellant's specification, paragraph [132]). This enables the query optimizer to partition data so as to minimize the amount of work performed in processing a query. The Examiner again refer to Srivastava (specifically, Srivastava, Section 3.1, paragraph 3 on page 86) as providing equivalent teachings. However, Appellant's review of that Section, as well as the balance of Srivastava, finds no teaching of a partitioning property such as that provided by Appellant's invention.

4. Claims 11, 36 and 57

Appellant's solution also provides for creating cost vectors for each operator tree and using these cost vectors during the scheduling process. As previously discussed, Appellant's invention considers both preemptable resources (PS) and non-preemptable

resources (NPS) during the schedule generation process (see e.g., Appellant's specification, paragraph [118]). More particularly, Appellant's claimed approach provides for representing costs of query operators as a pair of vectors with one dimension per PS and NPS resource, respectively (see e.g., Appellant's specification, paragraph [120]).

As to the above-described cost vector features included as limitations of Appellant's claims 11, 36 and 57, the Examiner again relies on Srivastava as providing equivalent teachings (see e.g., Final Rejection page 7 regarding Appellant's claim 11). However, as discussed above, while Srivastava acknowledges that a cost model is needed to account for operators being evaluated by multiple processors, it states that actual development of such cost expressions is a major task beyond the scope of the paper. Thus, Srivastava does not include the specific teachings of cost vectors comparable to those of Appellant's claimed invention.

5. Claims 12, 37 and 58

Appellant's claims 12, 37 and 58 includes claim limitations providing that the cost vector (of claims 11, 36 and 57) includes as components a selected one or more of work done by a processor in a given time, execution time of an operator, and resource usage of an operator for a certain time period (see e.g., Appellant's specification, paragraphs [114]-[115], paragraphs [123]-[129]). These cost vectors are used during scheduling and selection of optimal plans for execution of a query (Appellant's specification, paragraphs [114]-[115]). The Examiner references Srivastava as including equivalent teachings; however, the referenced teachings describe an analytical cost expression for evaluating a given relational operator of a different form (see e.g., Srivastava paragraph 3.1 at page 86). Moreover, it states that actual development of cost expressions of the form indicated for a given parallel architecture is beyond its scope. Thus, the teachings are not comparable to those of Appellant's claimed invention.

6. Claims 13, 38 and 59

Additionally, dependent claims 13, 38 and 59 add claim limitations providing that Appellant's optimization process includes pruning a first operator tree having a cost vector costing more in each vector dimension than a second operator tree (see e.g., Appellant's specification, paragraph [113], paragraph [129]). Srivastava does not include

any equivalent teaching as it does not mention the use of cost vectors nor does it include the more specific teachings of pruning of plans costing more in each vector dimension than other plans.

7. Conclusion

All told, the prior art references do not include teachings of adjusting operator trees (plans) based on resources (e.g., memory and threads) available for executing a given query. Additionally, the prior art references do not describe anything comparable to Appellant's scheduling process which generates a schedule for activation of operators of a particular plan based on dependencies among operators and available resources and which separates a given operator that is resource intensive into multiple operators when necessary as a result of resource limitations. Therefore as these references, even when combined, do not teach or suggest all of the claim limitations of Appellant's claims, it is respectfully submitted that claims 1-15, 17, 19-40, 43-62 and 64-70 distinguish over the combined references and the rejection under Section 103 should not be sustained.

B. Conclusion

Appellant's invention greatly improves the efficiency of the task of parallel execution of database queries. It is respectfully submitted that the present invention, as set forth in the pending claims, sets forth a patentable advance over the art.

In view of the above, it is respectfully submitted that the Examiner's rejection of Appellant's claims under 35 U.S.C. Section 103 should not be sustained. If needed, Appellant's undersigned attorney can be reached at 925 465 0361. For the fee due for this Appeal Brief, please refer to the attached Fee Transmittal Sheet. This Appeal Brief is submitted electronically in support of Appellant's Appeal.

Respectfully submitted,

Date: January 22, 2009

/G. Mack Riddle/

G. Mack Riddle; Reg. No. 55,572
Attorney of Record

925 465 0361
925-465-8143 FAX

8. CLAIMS APPENDIX

1. In a database system, a method for parallel optimization of a query, the method comprising:

generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;

adjusting parallel operators of each parallel plan based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable;

creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan, wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query and includes separating a resource intensive operator into a plurality of operators;

determining execution cost of each parallel plan based on its schedule; and

returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query.

2. The method of claim 1, wherein the query comprises a Structured Query Language (SQL) expression.

3. The method of claim 1, wherein said generating step includes generating an operator tree for each parallel plan based on the query.

4. The method of claim 3, wherein said step of generating an operator tree includes generating nodes of the operator tree as iterators for applying predefined behavior to data.

5. The method of claim 3, wherein said step of generating an operator tree includes inserting a parallel operator in the operator tree.

6. The method of claim 5, wherein said step of generating an operator tree includes dividing a query operation into sub-tasks and said parallel operator provides for executing said sub-tasks in parallel.

7. The method of claim 6, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of storage units.

8. The method of claim 6, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of CPUs.

9. The method of claim 5, wherein said parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

10. The method of claim 1, wherein said generating step includes generating a parallel plan using a partitioning property so as to partition data among operators of the parallel plan.

11. The method of claim 1, wherein said generating step includes generating a cost vector for each parallel plan.

12. The method of claim 11, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator in the parallel plan, and resource usage of an operator in the parallel plan for a certain time period.

13. The method of claim 11, wherein said generating step further comprises:

pruning a first parallel plan having a cost vector costing more in each vector dimension than a second parallel plan.

14. The method of claim 1, wherein said generating step includes generating a plurality of parallel plans based at least in part on partitioning and multi-dimensional

costing.

15. The method of claim 1, wherein said adjusting step includes adjusting a parallel plan based on maximum number of threads available at compile time.

16. (Canceled)

17. The method of claim 1, wherein said step of adjusting parallel operators of each parallel plan further comprises:

adjusting parallel operators based on available memory resources.

18. (Canceled)

19. The method of claim 1, wherein said creating step includes identifying pipelines in each parallel plan.

20. The method of claim 19, wherein said creating step includes constructing a pipeline dependency tree based on dependencies among operators of each parallel plan.

21. The method of claim 20, wherein said creating step includes determining order of execution of pipelines based on the pipeline dependency tree and available resources.

22. The method of claim 19, further comprising:

if resource usage of a particular pipeline is greater than resources available for the particular pipeline, splitting the particular pipeline into a plurality of pipelines.

23. The method of claim 22, wherein said step of splitting the particular pipeline includes adding operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

24. A computer-readable medium having processor-executable instructions for performing the method of claim 1.

25. A downloadable set of processor-executable instructions for performing the method of claim 1.

26. A system for parallel optimization of a database query, the system comprising:

a search engine for generating a plurality of parallel plans which can be used for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;

a parallel scheduler for adjusting parallel operators of each parallel plan based on maximum number of threads available for executing the query and creating a schedule for the parallel plan indicating a sequence for execution of operators of the parallel plan, wherein the maximum number of threads is user configurable and wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query and the parallel scheduler separates a resource intensive operator into a plurality of operators; and

a module for determining execution cost of each parallel plan based on its schedule, and returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query.

27. The system of claim 26, wherein the query comprises a Structured Query Language (SQL) expression.

28. The system of claim 26, wherein the search engine generates an operator tree for each parallel plan based on the query.

29. The system of claim 28, wherein the search engine generates nodes of the operator tree as iterators for applying predefined behavior to data.

30. The system of claim 28, wherein the search engine inserts a parallel operator in the operator tree.

31. The system of claim 30, wherein the search engine divides a query operation into sub-tasks and said parallel operator provides for executing said sub-tasks in parallel.

32. The system of claim 31, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of storage units.

33. The system of claim 31, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of CPUs.

34. The system of claim 30, wherein said parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

35. The system of claim 26, wherein the search engine generates a parallel plan using a partitioning property so as to partition data among operators of the parallel plan.

36. The system of claim 26, wherein the search engine generates a cost vector for each parallel plan.

37. The system of claim 36, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator in the parallel plan, and resource usage of an operator in the parallel plan for a certain time period.

38. The system of claim 36, wherein the search engine prunes a first parallel plan having a cost vector costing more in each vector dimension than a second parallel plan.

39. The system of claim 26, wherein the search engine generates a plurality of

parallel plans based at least in part on partitioning and multi-dimensional costing.

40. The system of claim 26, wherein the parallel scheduler adjusts a parallel plan based on maximum number of threads available at compile time.

41. - 42. (Canceled)

43. The system of claim 26, wherein the parallel scheduler identifies pipelines in the parallel plan.

44. The system of claim 43, wherein the parallel scheduler constructs a pipeline dependency tree based on dependencies among operators of a parallel plan.

45. The system of claim 44, wherein the parallel scheduler determines order of execution of pipelines based on the pipeline dependency tree and available resources.

46. The system of claim 43, wherein the parallel scheduler splits a particular pipeline into a plurality of pipelines.

47. The system of claim 46, wherein said parallel scheduler adds operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

48. A method for parallel optimization of a query requesting data from a database, the method comprising:

creating a plurality of operator trees for executing the query, the operator trees providing for execution of portions of the query in parallel;

adjusting the portions of the query to be executed in parallel based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable;

generating a schedule for execution of each operator tree based upon

dependencies among operators of each operator tree and resources available for executing the query including separating a resource intensive operator into a plurality of operators; and

returning a result indicating the operator tree having lowest execution cost based on its schedule for executing the query with available resources.

49. The method of claim 48, wherein the query comprises a Structured Query Language (SQL) expression.

50. The method of claim 48, wherein said creating step includes creating an operator tree including parallel operators for execution of portions of the query in parallel.

51. The method of claim 50, wherein said parallel operators comprise iterators for applying predefined behavior to data.

52. The method of claim 50, wherein said step of creating an operator tree includes creating operators for tasks to be performed in executing the query and said parallel operators provides for executing said tasks in parallel.

53. The method of claim 50, wherein a parallel operator executes in parallel across a plurality of storage units.

54. The method of claim 50, wherein a parallel operator executes in parallel across a plurality of CPUs.

55. The method of claim 50, wherein a parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

56. The method of claim 48, wherein said creating step includes creating an operator tree using a partitioning property so as to partition data among operators.

57. The method of claim 48, wherein said creating step includes generating a cost vector for each operator tree.

58. The method of claim 57, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator, and resource usage of an operator for a certain time period.

59. The method of claim 57, wherein said creating step further comprises:

pruning a first operator tree having a cost vector costing more in each vector dimension than a second operator tree.

60. The method of claim 48, wherein said creating step includes creating a plurality of operator trees based at least in part on partitioning and multi-dimensional costing.

61. The method of claim 48, wherein said adjusting step includes adjusting an operator tree for maximum number of threads available at compile time.

62. The method of claim 48, wherein said operator tree includes parallel operators for executing portions of the query in parallel and said adjusting step further comprises:

adjusting said parallel operators if necessary based on available memory resources.

63. (Canceled)

64. The method of claim 48, wherein said generating step includes identifying pipelines in each operator tree.

65. The method of claim 64, wherein said generating step includes constructing a

pipeline dependency tree based on dependencies among operators of each operator tree.

66. The method of claim 65, wherein said creating step includes determining order of execution of pipelines based on the pipeline dependency tree and available resources.

67. The method of claim 66, further comprising:

if resource usage of a particular pipeline is greater than resources available for the particular pipeline, splitting the particular pipeline into a plurality of pipelines.

68. The method of claim 67, wherein said step of splitting the particular pipeline includes adding operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

69. A computer-readable medium having processor-executable instructions for performing the method of claim 48.

70. A downloadable set of processor-executable instructions for performing the method of claim 48.

9. EVIDENCE APPENDIX

This Appeal Brief is not accompanied by an evidence submission under §§ 1.130, 1.131, or 1.132.

10. RELATED PROCEEDINGS APPENDIX

Pursuant to Appellant's statement under Section 2, this Appeal Brief is not accompanied by any copies of decisions.